Informe de Laboratorio Análisis y Diseño de Estructuras de Datos

Profesor: Cristian Sepúlveda S.
Alumno: Alberto San Martín C.

Fecha: 13 Noviembre 2023

Indice

Introducción

Los métodos de optimización tienen una actual vigencia en la informática moderna. Teorías y formulas para satisfacer soluciones a los problemas del dia de hoy y entregar una aproximación cercana y comprobada como optimizar recursos y obtener un mayor beneficio continua siendo un desafío útil para la humanidad. Es por cual, el presente informe intenta ser un marco teórico para introducir al lector como se utilizó y realizó la implementación del conocido método ***Knapsack Problem o problema de la mochila***. *Lo anterior con el fin de utilizarlo como formula para resolver la optimización de cargas de un buque de contenedores.*

Contexto y breve Descripción del Método de la Mochila.

El Problema de la Mochila (conocido también como ***Knapsack Problem*** o simplemente KP) es un problema clásico de la Investigación de Operaciones y en particular de la Programación Entera. Consiste en un excursionista que debe preparar su mochila, la cual tiene una capacidad limitada y por tanto no le permite llevar todos los artículos que quisiera tener en la excursión. Cada artículo que el excursionista puede incluir en la mochila le reporta una determinada utilidad. Luego el problema consiste en seleccionar un subconjunto de objetos de forma tal que se maximice la utilidad que el excursionista obtiene, pero sin sobrepasar la capacidad de acarrear objetos.

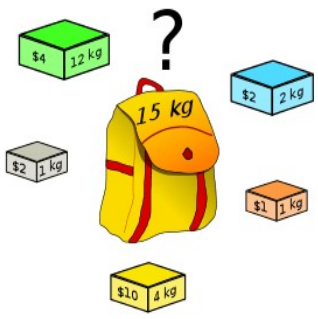
Problema actual donde se aplicara la metodología.

La compañía ***MAERSK S.A***. encargada del transporte marítimo de carga cuenta con una problemática la cual buscar una solución.

Debido a la fluctuación del dólar se hace necesario optimizar la entrega de cargas en sus buques contenedores. Lo anterior se refiere a que en cada viaje obtener un mayor costo de ganancia al momento de escoger el tipo de carga a transportar y por ende, que no sobrepase las capacidades de cada buque contenedor.

Es por lo anterior que se busco una formula para satisfacer dicha necesidad y se buscó en la academia los métodos mas comúnmente utilizados en el área de la investigación de operaciones.

Continuando con lo anterior, se utilizo el método KP (Knaspack Problem) para poder buscar una solución al problema de la transaccional **MAERSK S.A**. Lo anterior permitirá acercarnos a cual seria la carga mas optima a transportar que no sobrepase la capacidad del buque y permita obtener una mayor ganancia por producto transportado.



Propuesta de Solución basada en la Asignatura Estructura y Análisis de Datos en lenguaje C.

Para el problema señalado en puntos anteriores fue necesario de implementar un algoritmo y luego transformarlo a lenguaje C. Para lo anterior se describirá un algoritmo voraz que nos acerque a la solución del problema.

Algoritmo Knaspack

fuente: *https://pier.guillen.com.mx/algorithms/08-dinamica/08.6-mochila_cero-uno.htm#times-table*

```
01:  Var
02:     g,i,k,n,r,t,mw: longint;
03:     p,w: array [1..1000] of longint;
04:     c: array [0..1000,0..30] of longint;
05:
06:  Function max(a,b: longint): longint;
07:
08:  Function Knapsack(n, mw: longint): longint;
09:           var i,j: longint;
10:           begin
11:           fillchar(c, sizeof(c), 0);
12:           for i:= 1 to n do
13:               for j:= 1 to mw do
14:                   if (w[i]>j)
15:                      then c[i,j]:= c[i-1,j]
16:                      else c[i,j]:= max(c[i-1,j], c[i-1,j-w[i]]+p[i]);
17:           Knapsack:= c[n,mw];
18:           end;
19:
20:  Begin
21:       readln(input,t);
22:       for k:= 1 to t do
23:           begin
24:           readln(input,n);
25:           for i:= 1 to n do
26:               readln(input,p[i],w[i]);
27:           r:= 0;
28:           Knapsack(n,30);
29:           readln(input,g);
30:           for i:= 1 to g do
31:               begin
32:               readln(input,mw);
33:               Inc(r, c[n,mw]);
34:               end;
35:           writeln(output,r);
36:           end;
37:  End.
```

Descripción de Algoritmo Knaspack.

En las primeras cuatros líneas se declaran las variables que se utilizarán. Para los ciclos se utulizan i y k, las variables g, n, t, mw, p, w, las se utilizan de acuerdo a las especificaciones del problema, sólo que las dos últimas se construyen los arreglos. En la matriz c se guardaran los resultados intermedios de la mochila, en nuestro caso el contenedor y el resultado total se guarda en r.

La función max (línea 6) devuelve el menor de dos parámetros.

En la (líneas 8 a 18) se calcula el valor máximo que se puede guardar en la mochila o contenedor. Las variables i y j, se utilizan para los ciclos. Se comienza limpiando la matriz c (línea 11). Para cada objeto de la mochila o contenedor (línea 12), se va calculando el valor máximo que se puede obtener con un peso máximo de j (líneas 13 a 16). Al final se regresa al valor máximo que se puede obtener utilizando todos los objetos (línea 17).

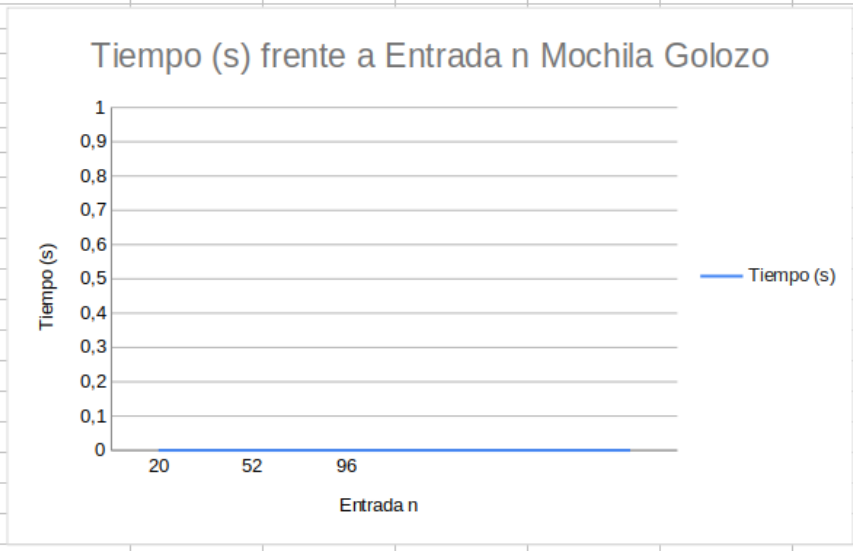
Entre las líneas 20 y 37 se tiene el bloque principal del programa. Comienza leyendo la cantidad de casos a procesar (línea 21). Para cada caso, se lee el número de objetos (línea 24), junto con el valor y peso de estos (líneas 25 y 26). Después se inicializa el resultado y se calcula la mochila con el valor máximo (30). Se la cantidad de filas y (línea 29) y para cada una de ellas, se lee el máximo peso que pueden cargar (línea 32) y se incrementa el resultado con el valor máximo que se pueden obtener (línea 33). Se termina escribiendo el resultado (línea 35).

Tiempos de Ejecución Algoritmo Golozo – Mochila.

Los tiempos calculados para la ejecución del algoritmo Knaspack para la solución del problema de los contenedores tuvo los siguientes tiempos. Se pueden ver en la figura.

La ejecución de una entrada de 96 registros se encontró muy por debajo se un segundo. Se podrían realizar pruebas con entradas mas amplias y así verificar su rendimiento.

Mochila/Contenedor Golozo	
Entrada n	Tiempo (s)
8	0.000049
20	0.000127
52	0.000802
96	0.001345



Conclusión

Se puede observar que el algoritmo tiene tiempos muy bajos para entradas de datos pequeñas. Para lograr un mayor efecto seria necesario probar con gran cantidad de datos de entrada. Sin embargo se puede observar que el lenguaje de programación C continua siendo de muy rápida ejecución.